METHOD OF SUPPLYING CONTENT TO A DEVICE

FIELD OF THE INVENTION

The present invention relates to a method of supplying content to a device and in particular to a method of supplying content to mobile devices using a mobile communications network.

BACKGROUND OF THE INVENTION

One of the growth areas for mobile network operators and content providers is the provision of ringtones, wallpapers and other multimedia content for mobile telephones and devices. There is a tension between the needs of mobile network operators and device manufacturers to retain control over some aspects of the device user interfaces for branding purposes and the needs of users to customise and modify the appearance of their devices to suit their own needs. The sophisticated software required to provide the desired flexibility and customisation is also in tension with the limited processing power and data storage capacity of typical mobile devices. The present invention seeks to mitigate these problems.

SUMMARY OF THE INVENTION

According to a first aspect of the present invention there is provided a method of receiving content data for a user interface to a device, the method comprising the steps of: the device receiving content data for a user interface from a communications interface; the device processing the received

content data to form a user interface for the device; wherein the content data comprises metadata and the method comprises the further step of the device accessing content data updates via the communications interface in accordance with the content data metadata.

According to a second aspect of the present invention there is provided a data carrier comprising computer-executable code for performing the above-described method.

According to a third aspect of the present invention there is provided a device comprising: a user interface; a display means for displaying the user interface; a communications interface for receiving content data for use in the user interface and processing means to process received content data to form the user interface wherein the content data comprises metadata and the device is configured to access content data updates via the communications interface in accordance with the content data metadata.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 shows a schematic depiction of a system incorporating the present invention;

Figure 2 depicts in greater detail the structure and operation of server 100;

Figure 3 shows a schematic depiction of the software for the mobile devices;

Figure 4 shows a schematic depiction of the content toolset 200; and

Figure 5 shows a schematic depiction of a device that comprises a user interface according to an embodiment of

the present invention.

DETAILED DESCRIPTION OF THE INVENTION

The invention will now be described, by way of illustration only, with respect to the accompanying drawings, in which Figure 1 shows a schematic depiction of a system incorporating the present invention. The system comprises server 100, content toolset 200, mobile devices 300, operational support systems (OSSs) 700, content feeds 500 and user interface (UI) sources 600. In use, the server 100 communicates content data and UI data to the mobile devices 300, 301, ..., each of which comprise software package 400. The server 100 interfaces with OSSs 700, with the OSSs being those conventionally used to operate mobile networks, for example billing, account management, etc. The server 100 further interfaces with the content toolset 200: the content toolset receives data from UI sources 600, 601, ..., and packages the UI data such that the server can transmit the packaged UI data to the software packages 400 comprised within the mobile devices 300. The server receives data from a plurality of content feeds, and this data is processed and packaged such that it can be sent to the software packages 400 or so that the mobile devices 300 can access the data using the software package 400.

The system can be envisaged as being divided into three separate domains: operator domain 50 comprises the systems and equipment operated by the mobile network operator (MNO); user domain 60 comprises a plurality of mobile devices and third-party domain 70 comprises the content feeds and UI feeds that may be controlled or operated by a number of

different entities.

Figure 2 depicts in greater detail the structure and operation of server 100. Server 100 comprises publishing component 110 and content server component 150. Publishing component comprises database 111, import queue 112, content toolset interface 113, user interface 114 & catalogue 115. In operation, the publishing component receives content from the content toolset at the content toolset interface. The content is presented in the form of a parcel 210A, 20B, …,(see below) comprising one or more Trigs and one or more Triglets. A trig is a user interface for a mobile device, such as a mobile telephone and a triglet is a data file that can be used to extend or alter a trig. If a parcel comprises more than one trig then one of the Trigs may be a master trig from which the other Trigs are derived.

The publishing component user interface 114 can be used to import a parcel into the database 111, and this process causes references to each trig and triglet to be loaded into the import queue 114, which may comprise references to a plurality of parcels 210a, 210b,… . The contents of the parcel may be examined using the user interface and the contents of the parcel can be passed to the catalogue.

The MNO may have several publishing domains, for example one for each target server in a number of countries or regions. Each domain is treated in isolation from other domains and has its own publishing scheme describing how objects are to be published onto content servers in both live and staging environments. The publishing component GUI provides several different views to each domain, enabling operators to

completely manage the publishing of content. The catalogue comprises references to the Trigs stored in the catalogue and the update channels and feed channels used to transfer content to the various domains. For each domain, the operator uses the publishing component GUI to set up the domain structure and allocate trigs from the catalogue to each domain node. To aid the operator in selecting trigs efficiently, a filter is provided in the catalogue so that only relevant items are shown.

A trig may be allocated to several nodes within a domain. In each case the packaging of the trig on the target server may need to be different e.g. a SIS or CAB file, dependent on the handsets that will be accessing the trigs. The packaging can be controlled using the publishing component GUI.

The update channels may be referenced by trigs to control the delivery of the content. An update channel comprises a URL which is a link to a resource on the associated domain that comprises a triglet update package. The URL can be polled at predefined intervals and the HTTP GET function used to access the package (it will be readily appreciated that other transport schemes may be used, for example SyncML or SMS or cell broadcast for small updates). The triglet update package describes how the trig can be modified, e.g. replacing one or more images or text files used by the trig. The publishing component GUI enables an operator to define and control the update channels that exist for a domain, the URLs associated with each triglet on the update channel and the association of triglets with the update channels for a domain. As each triglet is associated with an update channel, an operator may enter the date and time that the

update should be published, enabling a schedule to be set.

A content feed is accessed by polling a URL, retrieving the update packet and applying it to the trig. However because of the different nature of manually constructed triglet updates and automatically generated content, update channels and content feeds are managed separately. Again, other transport schemes may be used such as SyncML or OMA-DM (Open Mobile Alliance Device Management).

The publishing component GUI enables the operator to define which content feeds should be available within each domain and a platform specific location identifier, for example a URL, to which the content should be posted. The operator defines content feeds themselves using a separate screen within the publishing component GUI. All domain publishing scheme information at this stage is held in the database.

If an existing live Domain is to be modified, the publishing component operates on a copy of the live domain structure, and defines the changes to be made to the domain e.g. the removal or addition of trigs. The individual trigs or triglets allocated to the publishing scheme reference the parcels from which they were originally imported. This enables the publishing component to compile them later (see below).

In an alternative embodiment, the update packet can be accessed in a different manner. An interface can be provided to the database 111 such that the interface can be polled by the device in order to request a content update. The database holds an entry for the location of the relevant

content update packet, which is then returned to the device. The device is then able to access the content update packet by polling the appropriate URL. This alternative approach significantly simplifies the management and operation of the content feeds as it is no longer necessary to identify the location of updates when content is first issued to terminals and provides a degree of automation to the system. Content updates may be located (and re-located) as required without needing to notify all of the devices of the changes, as long as the database is updated.

The operator is able to select different views on the Domain under development, for example the original structure (i.e. what is currently live), a final proposed structure (Approved and/or Pending items), with or without changes marked, the changes only and rejected items. The publishing component GUI prevents a domain scheme from being published if it contains trigs or triglets which reference update channels or content feeds that have not yet been allocated to the domain. Once a publishing scheme is ready to be tested, it is published to the domain's staging server for testing.

The publishing request is processed by the server and comprises compiling all uncompiled Trigs and Triglets (both Approved and Pending) and exporting all proposed changes, both pending and approved, to the Staging Server (this includes new trigs, updates to existing trigs, triglets overdue for publishing (according to a test date) and removal of trigs, triglets and nodes. If there are any failures at compilation stage, no items will be published. The offending item must be rejected or corrected to allow publishing to continue.

Once on the MNO's staging server, the Domain can be tested using mobile devices. Each item that passes its test can be marked as "Approved" using the publishing component GUI. Items that do not pass tests can be marked as "Rejected". Corrected Trigs/Triglets can be imported into publishing component, resubmitted to the Domain and then published to the staging server as described above. This process continues until all items are approved and the Domain is ready for publishing to the live environment. Additionally, for the staging area of a domain, it is possible to simulate the passage of time so that scheduled updates can be tested. Some MNOs may not need the Staging server capability and thus all items to be published can be marked as approved when the domain scheme is set up and thus the operator can proceed directly to live publishing.

The publishing component GUI provides views of each of the domains, for both Live and Staging areas. From this view it is possible to start and stop automatic publishing of content feeds and scheduled publishing of triglets on each update channel.

Having completed testing, the domain may be published to the live area of the server, using a process similar to that described above, except that only domain changes marked as approved are published. When setting up a publishing scheme, the dates and times at which to publish individual Trigs and Triglets may be set. On requesting that a domain is ready for publishing, publishing component ensures that all trig and Triglets are compiled - even if the publishing date or time associated with the item is in the future. Future dated items

remain in the publishing component database until they are ready to be dispatched to the target domain area. A scheduled publish process may poll each target Domain area at configured intervals and dispatches any scheduled items which are due or overdue for publishing to the appropriate Domain target via the dispatch API.

Content feeds are updated at regular intervals with dynamic content being extracted from external sources e.g. web scraping, RSS news feeds, etc. The dynamic content may be simple news ticker text (headlines and URLs) but could also include more complex objects of the type that might be replaced using the content toolbox. The content feed process formats this content into an update file and then passed back to Feed Control.

Feed Control invokes the Compile Trigs process which passes the triglet Parcel template associated with the Content Feed and the update file to the compiler. The compiler extracts the resources in the triglet parcel and returns a compiled triglet to Feed Control which can then be published (see above).

For each trig or triglet to be compiled, the complier requires the following information: the original parcel in which the trig/triglet was imported or created; the list of trig/triglet update packages to be created; the type of files to be created; and a URL map for the update channels and content feeds. The compiler uses the URL map to update the URLs referenced by the update channels (and content feeds) within the individual trigs and triglets within each parcel.

A dispatch API may be used for dispatching content to the MNO's servers. A reference FTP model has been implemented to service the API and transfer files to a content server however the API mechanism enables an integrator to implement their own content dispatch mechanism for example using the publishing component output as input to the API of their own content management system, adding custom logic if required.

The publishing component supports conventional OSS functionality accessible via the publishing component GUI and via an industry standard API (JMX) which enables SIs to use standard integration tools to integrate the publishing component into the MNO's OSS environment. This includes the logging of significant publishing component events and all imported/published items, an audit trail for any changes noted with external scripts, maintain Error/Warning Logs, system alerts, health check report, etc. All data relating to the publishing component is stored within a database, such as Oracle, and backup and restore functionality is supported by the standard database processes, integrated with the OSS environment. For operation the publishing component requires an operating J2EE environment and an installed instance of a database, such as Oracle. Installation of the publishing component can be validated by a process that indicates that that the installation process was successful and that all components have been activated correctly.

The content server component 150 is a standard implementation of a web server and as such the scaling model is well understood. The capabilities of a server can be rated using a "SPECweb99" number indicating the number of concurrent sessions that the web server can handle under benchmark

conditions. Published SPECweb99 numbers range from 404 to 21,000 with typical commercial web servers having SPECweb99 numbers in the order of 5,000. A typical deployment scenario of 1 million subscribers with hourly updating content requires a web server with a SPECweb99 rating of only 1,112. A successful deployment will lead to increased service use which can be provided for by enabling additional servers to create an infrastructure that can be both scalable and highly resilient to failure.

A connection may be made to the server from a mobile device via a WAP Gateway. In this case the web server session exists between the WAP gateway and the web server, rather than the mobile phone and web server. When a request is made for a file via the WAP gateway, the session with the web server lasts only as long as it takes to transfer the file from the web server to the WAP gateway - i.e. the session is extremely short since the connection bandwidth will be very high and latency extremely low.

Alternatively a direct connection may be established between the mobile phone and the web server. In this case, the web server will need to keep the session open for as long as it takes to download the data to the phone.

There are two types of content that are delivered by the content server component: trigs, typically of the order of 100KB and regularly updating triglets which are typically of the order of 1KB. The traffic created by trig downloads is very similar to the traffic generated by existing content downloads. And thus the related issues are well understood. Downloads of regular triglet updates are a new feature in an

MNO's traffic model but because of the small size of the updates, which typically fit within one data packet, it is possible to show that the traffic can still be handled by typical web servers.

In the case of a triglet download, typically only one data packet is required to transfer 1KB. Assuming a round-trip latency across a GPRS network of 2 seconds, the web server will need to hold open a typical session for around 4 seconds. For the scenario of 1 million subscribers having a trig on their phone with content that updates every hour, this implies 278 hits per second on the web server and 1,112 concurrent sessions. As stated earlier, this number is well within the capability of typical web servers.

Figure 3 shows a schematic depiction of the software 400 for the mobile devices 300, which comprises a mark-up language renderer 410, update manager 420, network communication agent 425, resource manager 430, virtual file system 435, actor manager 440, a plurality of actors 445a, 445, ..., native UI renderer 450, support manager 460, trig manager 465 and mark-up language parser 470.

The software may operate using TrigML, which is an XML application and that mark-up language renderer 410 renders the TrigXML code for display on the mobile device 300. TrigML is described in our co-pending application GB0403709.9, filed February 19th 2004, the contents of which are incorporated herein by reference.

The mark-up language renderer also uses the TrigML Parser to parse TrigML resources, display content on the device screen

and controlling the replacement and viewing of content on the handset. The native UI renderer is used to display UI components that can be displayed without the use of TrigML, and for displaying error messages.

The software 400 is provisioned and installed in a device specific manner. Similarly, software upgrades are handled in a device specific manner. The software may be provisioned in a more limited format, as a self-contained application that renders its built in content only: i.e. the software is provisioned with a built-in trig but additional trigs cannot be added later. The supplied trig may be upgraded over the air.

The resource manager provides an abstraction of the persistent store on device, i.e. storing the files as real files, or as records in a database. The resource manager presents a file system interface to the mark-up language renderer and the update manager. It is responsible for handling file path logic, distinguishing between real resource files and actor attributes, mapping trig-relative paths onto absolute paths, interfacing with the trig manager and providing a modification interface to the update manager.

The Update Manager handles the reception and application of Trigs and Triglets. The Update Manager presents an interface to the Renderer and the trig Manager and is responsible for: the initiation of manual updates when instructed to by the Renderer; controlling and implementing the automatic update channel when so configured by the trig manager; indicating the progress of a manual update and recovering an Update following unexpected loss of network connection and/or device

power. The update packet format may be defined as a binary serialisation of an XML schema.

The Support Manager provides an interface for other components to report the occurrence of an event or error. Depending on the severity of the error, the Support Manager will log the event and/or put up an error message popup.

The Actor Manager 440 looks after the set of actors 445 present in the software. It is used by: the renderer when content is sending events to an actor; actors that want to notify that an attribute value has changed and actors that want to emit an event (see below).

The software 400 is provisioned to mobile devices in a device specific method. One or more Trigs can be provisioned as part of the installation, for example, stored as an uncompressed update packet. On start-up, the packet can be expanded and installed to the file-system.

The Actors 445 are components that publish attribute values and handle and emit events. Actors communicate with the Renderer synchronously. If an actor needs asynchronous behaviour, then it is the responsibility of the actor to manage and communicate with a thread external to the main thread of the Renderer.

Actor attributes may be read as file references. Attributes are one of four types: a single simple value; a vector of simple values; a single structure of fields, each field having a simple value; or a vector of structures. Attributes may be referenced by an expression using an object.member

notation similar to many object-orientated programming languages:

```
<image res="signallevels/{protocol.signalstrength}"/>
```

When needed as a file, an attribute is accessed via the /attrs folder.

```
<text res="/attr/network/name">
```

An Actor can be messaged by sending it an event with the <throw> element. Events emitted by actors can be delivered to the content tree as content events: these can be targeted at an element Id or 'top'. The interface to an actor is defined by an Actor Interface Definition file. This is an XML document that defines the attributes, types, fieldnames, events-in and parameters, and events out. The set of actors is configurable at build-time for the software. Appendix A gives an exemplary listing of some actors that may be used, along with the associated functions or variables.

Updates comprise a new trig (a new or replacement UI) or a triglet (a modification to an existing trig) and may be regarded as modifications to the software file-system. The Update Manager to determine what needs changing in the file-system by reading a packet. Update Packets may be downloaded over the air by the software 400 using HTTP, or other suitable transport mechanisms, wrapped in a device-specific package format or pre-provisioned with the installation of the software itself.

Updates may be triggered by a number of means, which include

- the software checking for interrupted Update processing on start-up
- the software checking for pre-installed Update Packets on start-up
- automatically as configured by an Update Channel
- user initiation
- the device receiving a special SMS

Updates sent OTA can be fetched using HTTP-GET requests initiated by the software. The GET request is directed at the URL associated with the Update. The body of the HTTP response is a binary file carrying data in an Update Packet format. The data reception is handled in a separate thread to the Renderer thread. For background updates (automatically initiated) this allows the user to continue navigating the UI. For foreground updates (manually initiated) this allows the Renderer thread to display a progress bar and listen for a cancel instruction.

Trigs may define Update Channels, which comprise a URL from which to poll for updates and a number of associated properties: automatic or manual updating; a timing scheme and retry strategy in the event of failure. The software may only initiate an automatic update when it is running.

On start-up, if an update event is due the software will wait for a time interval before beginning the Update. This is to postpone any start-up delays incurred by initiating an Update immediately, and will therefore give the user a faster start-up. The update channels may be defined in a well known location within the trig, for example the config/channels folder of a trig in files containing <channel> tags.

The algorithm used to unpack and install an update is device specific. However, it is important that the algorithm is safe from unexpected interruption (e.g. power loss), such that no corruption or unrecoverable state is reached in the file-system. This may be achieved by using two threads (a network thread and a renderer thread) with the goal of having as much of the update processing as possible being performed by the network thread so as to interrupt the renderer thread for the shortest possible amount of time.

There are other failure modes to consider: if an HTTP-GET cannot be initiated, or is met with an HTTP error response code, then this attempt at an Update is abandoned and the retry strategy is used to begin a new update attempt at a later date. Where an HTTP response is interrupted by loss of network signal, any temporary files are deleted and the retry strategy is used to restart the Update attempt at a later date. If an update header indicates that the update payload size may be too great to fit on the device, if the update requires an incompatible version of the software or if the Update already resides on the device then the header data file is deleted and the Update attempt and any subsequent retries are cancelled.

The content format is common across all platforms implementing the software. The Content Compiler is a content authoring tool to transform a collection of raw resources (text TrigML, PNG images, text string definitions) into an over the air Update Packet that can be written to the file system of the device.

It is desirable that the cost of re-branding UIs and producing a continual stream of updates is minimal. This is enabled this by providing an efficient flow of information from the creative process through to the transmission of data to users.

A container, referred to as a parcel, is used for UIs, UI updates, and templates for 3rd party involvement. Parcels contain all the information necessary for a 3rd party to produce, test and deliver branded UIs and updates. Figure 4 shows a schematic depiction of the content toolset 200, which comprises scripting environment 220, test and simulation environment 230 and maintenance environment 240

The parcel process comprise five processing stages:

1) The scripting environment 220 provides the means to design the template for one or more UIs and the update strategy for UIs based on that template.

2) The maintenance environment 240 provides for rapid UI and update production in a well-controlled and guided environment that can be outsourced to content providers.

3) The maintenance environment 240 'pre-flight' functionality allows the deployment administrator to check and tune the UIs and updates that they receive from 3rd parties.

4) The publishing component 110 provides management of UIs and updates at the deployment point, including the staging of new releases.

5) The publishing component 110 enables the automatic generation of updates from live content feeds.

In a typical project, parcels are created within the scripting environment 220 for: a content provider to create

re-branded UIs from a template, incorporating the same 'feel' but a different 'look'; a content provider to create updates from a template, that provide a periodic, or user selected variation to UI content ; or an ad agency to create updates from a template that promote new services on a periodic basis.

For all of these use cases, maintenance environment 240 is used to import the parcel, re-brand and reconfigure the content and create a new parcel for submission to the publishing component 110. In the design of the UI template, the following issues should be considered: which part of the UI can be re-banded; which features of a UI need to be reconfigured at re-branding or remotely; which part of the UI content may be updated; and if the UI is re-branded then can user select content feeds in use. The scripting environment 220 allows these strategies to be defined, and enables the maintenance environment 240 as the implementer of each instance of each strategy.

The parcel format is an opaque binary format that stores all this information as serialized objects. The parcel may comprise a number of resources , such as images, text, URLs, update channels, ringtone files, wallpapers, native applications, etc. Each resource contains permission information as to how to view, edit, or delete the resource. Each resource furthermore contains meta information such as documentation and instructions that are relevant to that resource. Each Parcel tool either assumes a relevant role, or requires users to login as a particular role.

Figure 5 shows a schematic depiction of a device 800 that comprises a user interface according to an embodiment of the present invention. The device comprises a display 810 that displays the user interface 815 and user interface means 820, that enable the user to interact with the user interface 815. A processor 830 executes the software that is stored within one or more storage means 840 and there may be provided one or more wireless communication interfaces 850, to enable communication with other devices and/or communication networks. One or more batteries 860 may be received to power the device, which may also comprise interfaces to receive electrical power and/or communication cables.

The nature of these components and interfaces will depend upon the nature of the device. It will be understood that such a user interface can be implemented within a mobile or cellular telephone handset, but it is also applicable to other portable devices such as digital cameras, personal digital organisers, digital music players, GPS navigators, portable gaming consoles, etc. Furthermore, it is also applicable to other devices that comprise a user interface, such as laptop or desktop computers.

The user interface means may comprise a plurality of buttons, such as a numerical or alpha-numerical keyboard, or a touch screen or similar. One or more storage devices may comprise a form of non-volatile memory, such as a memory card, so that the stored data is not lost if power is lost. ROM storage means may be provided to store data which does not need updating or changing. Some RAM may be provided for temporary storage as the faster response times support the caching of frequently accessed data. The terminal may also accept user

removable memory cards and optionally hard disk drives may be used as a storage means. The storage means used will be determined by balancing the different requirements of device size, power consumption, the volume of storage required, etc.

Such a device may be implemented in conjunction with virtually any wireless communications network, for example second generation digital mobile telephone networks (i.e. GSM, D-AMPS), so-called 2.5G networks (i.e. GPRS, HSCSD, EDGE), third generation WCDMA or CDMA-2000 networks and improvements to and derivatives of these and similar networks. Within buildings and campuses other technologies such as Bluetooth, IrDa or wireless LANs (whether based on radio or optical systems) may also be used. USB and/or FireWire connectivity may be supplied for data synchronisation with other devices and/or for battery charging.

Computer software for implementing the methods and/or for configuring a device as described above may be provided on data carriers such as floppy disks, CD-ROMS. DVDs, non-volatile memory cards, etc.

This application claims the benefit of UK Patent Application number 0403709.9, filed February 19th 2004, the contents of which are incorporated herein by reference.

APPENDIX A

Trigplayer Actor	Attributes	UpdateState	
	Messages	exit	
		predial_mode	on/off
	Events	idle	

Launch Actor	Attributes		
	Messages	browser	url
		SMS	Number
			message
		Camera	
		Inbox	
		Profiles	
		missed_calls	
		dialer	number
		...	
		native_app	app_id
			url
	Events		

Install Actor	Attributes		
	Messages	ringtone	resource_path
		wallpaper	resource_path
	Events		

Phone Actor	Attributes	Bluetooth	
		IrDA	
		Call	
		GPRS	
		UnreadSMS	
		UnreadVoiceMail	
		UnreadMsgs	
		BatteryLevel	
		SignalStrength	
	Messages		
	Events	missed_call	
		message_arrived	
		voice_mail_arrived	